Requested Patent: WO9419748A2

Title:

METHOD OF TRANSFERRING DATA USING DYNAMIC DATA BLOCK SIZING ;

Abstracted Patent: WO9419748 ;

Publication Date: 1994-09-01 ;

Inventor(s): DONALD JAMES A;; TEDDY JOHN D ;

Applicant(s): CENTRAL POINT SOFTWARE INC (US) ;

Application Number: WO1994US00337 19940111 ;

Priority Number(s): US19930002635 19930111 ;

IPC Classification: G06F13/00 ;

Equivalents: AU7090794 ;

ABSTRACT:

An improved method of transferring data among computer system Input/Output devices by an application running under a nonpreemptive multitasking operating system. Data is transferred in blocks between Input/Output devices wherein the blocks are continually and dynamically re-sized during the transfer of the data. The method further includes the step of relinquishing control of the computer system central processing unit between each transfer of a data block whereby performance of the computer system is improved. The method achieves relatively uniform response time independent of the speed of operation of the Input/Output device.

modifies size of packets, not size of memory (memory allocation)

1748

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 13/00 | A2

(11) International Publication Number: WO 94/19748

(43) International Publication Date: 1 September 1994 (01.09.94)

(21) International Application Number: PCT/US94/00337

(22) International Filing Date: 11 January 1994 (11.01.94)

(30) Priority Data:
002,635 11 January 1993 (11.01.93) US

(71) Applicant: CENTRAL POINT SOFTWARE, INC. [US/US]; Suite 200, 15220 N.W. Greenbrier Parkway, Beaverton, OR 97006 (US).

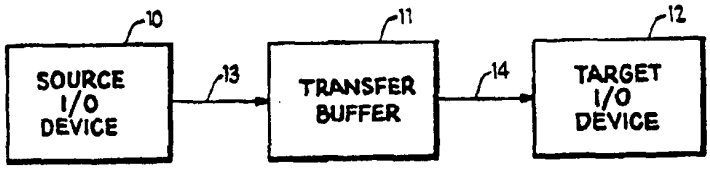
(72) Inventors: TEDDY, John, D.; 908 Corsair Lane, Foster City, CA 94404 (US). DONALD, James, A.; 1068 Fulton Avenue, Sunnyvale, CA 94089 (US).

(74) Agents: DICK, Richard, Eugene et al.; The Law Offices of Dick and Harris, 181 West Madison Street, Suite 3800, Chicago, IL 60602 (US).

(81) Designated States: AU, CA, JP, KR, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: METHOD OF TRANSFERRING DATA USING DYNAMIC DATA BLOCK SIZING



(57) Abstract

An improved method of transferring data among computer system Input/Output devices by an application running under a nonpreemptive multitasking operating system. Data is transferred in blocks between Input/Output devices wherein the blocks are continually and dynamically re-sized during the transfer of the data. The method further includes the step of relinquishing control of the computer system central processing unit between each transfer of a data block whereby performance of the computer system is improved. The method achieves relatively uniform response time independent of the speed of operation of the Input/Output device.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**GB**	United Kingdom	**MR**	Mauritania
AU	Australia	**GE**	Georgia	**MW**	Malawi
BB	Barbados	**GN**	Guinea	**NE**	Niger
BE	Belgium	**GR**	Greece	**NL**	Netherlands
BF	Burkina Faso	**HU**	Hungary	**NO**	Norway
BG	Bulgaria	**IE**	Ireland	**NZ**	New Zealand
BJ	Benin	**IT**	Italy	**PL**	Poland
BR	Brazil	**JP**	Japan	**PT**	Portugal
BY	Belarus	**KE**	Kenya	**RO**	Romania
CA	Canada	**KG**	Kyrgystan	**RU**	Russian Federation
CF	Central African Republic	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CG	Congo			**SE**	Sweden
CH	Switzerland	**KR**	Republic of Korea	**SI**	Slovenia
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SK**	Slovakia
CM	Cameroon	**LI**	Liechtenstein	**SN**	Senegal
CN	China	**LK**	Sri Lanka	**TD**	Chad
CS	Czechoslovakia	**LU**	Luxembourg	**TG**	Togo
CZ	Czech Republic	**LV**	Latvia	**TJ**	Tajikistan
DE	Germany	**MC**	Monaco	**TT**	Trinidad and Tobago
DK	Denmark	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**US**	United States of America
FI	Finland	**ML**	Mali	**UZ**	Uzbekistan
FR	France	**MN**	Mongolia	**VN**	Viet Nam
GA	Gabon				

Title Of The Invention:

Method Of Transferring Data Using Dynamic Data Block Sizing

Background of the Invention

1. Background Art

The present invention relates generally to a method for transferring data in blocks from one input/output (I/O) device to a second I/O device within a computer system, and in particular to a method for dynamically sizing said data blocks to improve the overall performance and responsiveness of nonpreemptive multitasking operating systems to user input.

2. Technical Field

The operating system for "XT" class personal computers initially offered by International Business Machine Corporation, and compatible computers, comprised the DOS operating system originating with Microsoft Corporation, which did not provide for multitasking such that the user could not freely switch from application to application without ceasing operation of the first application before switching to the second. Recently, nonpreemptive multitasking operating systems for "386" and "486" class IBM, and compatible, personal computers have become increasingly popular. An example of such an operating system comprises the combination of MS-DOS 5.0 and WINDOWS 3.1, both from Microsoft Corporation of Redmond, Washington. Nonpreemptive multitasking operating systems differ from true preemptive multitasking operating systems in that they do not employ time-slicing to allocate Central Processing Unit

(CPU) time to concurrently running applications. Rather, an application that currently has control of the CPU retains such control until it is voluntarily relinquished. All other concurrently running applications are suspended until the currently running application relinquishes CPU control. Under certain circumstances and when performing certain tasks non-preemptive operating systems may experience slow system responsiveness wherein the user is forced to wait for the computer system to complete one task before the user is permitted to regain control of the system or even have commands entered using a keyboard or mouse input device acknowledged.

An example of a task performed by such an application is the transfer of data from one Input/Output (I/O) device to another, such as the copying of a data file from a hard disk drive to a floppy disk drive. Slow system responsiveness to user inputs often occurs when an application releases CPU control too infrequently. System responsiveness may also be slowed by the operating system overhead time involved in frequent task-switching. This can occur when an application performing a lengthy operation relinquishes and then recovers CPU control too frequently, by subdividing the operation into too many portions.

In certain prior art operating systems and applications, the task of copying a data file from one I/O device to another I/O device is accomplished by transferring blocks of data of a fixed size. Where the source device is a relatively high performance device capable of speedy data transfer but the data is transferred in relatively small blocks, many more 'reads' will be required to completely transfer the entire data file. Moreover, where certain operating

systems and applications do not relinquish control of the CPU until the entire file transfer is complete, the user is effectively 'locked-out' and unable to perform any operations until the task is complete. The copying of files of sufficient size would thus cause the user to wait until the copying is done before resuming use of the computer. Printing is another example of the transfer of data which may effectively 'lock-out' the computer user.

Certain other prior art applications, including many Windows 3.1 applications, 'monopolize' CPU time when transferring data among Input/Output (I/O) devices or memory, albeit to a lesser degree. These prior art applications relinquish CPU control after every transfer of a fixed-sized portion, such as a 64 Kilobyte (K) block, of the data to be transferred among I/O devices, as opposed to retaining control until the entire data file is transferred. While this method is an improvement over prior art applications that totally monopolize CPU time when transferring a data file, its use of relatively large fixed-size data blocks can still result in slower system responsiveness to user input. Applications running under a nonpreemptive multitasking operating system cannot interrupt their I/O device accesses in midstream in order to return CPU control to the operating system. Instead, an I/O device access, once initiated, must be completed before CPU control can be relinquished by the application. As a result, the time during which such prior art applications will maintain CPU control before relinquishing CPU control to the operating system will vary widely, depending upon the actual data access time of the particular I/O device being accessed at any given moment.

For example, where data is being transferred to slow I/O devices, such as floppy disk drives, system performance may be adversely affected by transferring files using a large fixed data block size, since control of the CPU would be released too infrequently. Since the data access times of I/O devices are proportional to the amount of data to be transferred within a given access, the transfer of a large block of data to or from a relatively slow I/O device will result in a noticeable degradation of the overall responsiveness of the computer system.

Accordingly, it is an object of the present invention to provide a method of dynamically modifying the size of data blocks transferred among I/O devices in order to improve the responsiveness of nonpreemptive multitasking operating systems to user input.

It is a further object of the present invention to modify the size of data blocks transferred among I/O devices as a function of the performance of the I/O devices as measured by time taken to transfer the immediately preceding data block.

It is additionally an object of the present invention to provide such a method of dynamically modifying data block size that includes the use of initial block size values based upon the estimated speed of I/O devices.

These and other objects of the present invention will become apparent in light of the present specification and drawings.

Disclosure Of The Invention

The present invention comprises a method for the transferring of data among Input/Output (I/O) devices as commanded by an application operating within a nonpreemptive multitasking computer operating system environment, such as the combination of MS-DOS 5.0 and Microsoft Windows 3.1 from Microsoft Corporation. Dynamically sized data blocks are transferred by reading and then writing the data in a manner which optimizes the overall time and uniformity of the time an application retains CPU control when performing data transfers towards improving the overall responsiveness of the operating system and concurrently running applications to user input.

An application employing the method of the present invention performs data transfers among I/O devices or memory by partitioning the total amount of data to be transferred into smaller discrete blocks. Blocks of data are transferred from a source I/O device into a transfer buffer of fixed, predetermined size allocated within a computer system's Random Access Memory (RAM). The initial size of the read block is pre-selected according to the estimated speed of the type of I/O device or memory being accessed. Traditionally slow source I/O devices, such as floppy disk drives, are initially assigned read blocks of a relatively small size. Memory and traditionally fast source I/O devices, such as hard disk drives, are initially assigned read blocks of a relatively large size. As such initial estimates are not always an accurate prediction of the performance of an I/O device, as measured by the actual speed of read accesses to a given I/O device, and

accordingly, the method of the present invention includes dynamic sizing to adjust the size of read blocks to the coincide with the actual initial performance of I/O devices as well as the varying performance of the I/O devices.

For every read operation, the time required to read a block of data from the source I/O device is determined. To perform the transfer of data an initial block of data of a preselected size is read into the transfer buffer. If a given read operation takes longer than is desirable, for instance longer than approximately 700 milliseconds (ms), the size of block for the next read operation is made smaller in order to reduce the time required for the next read operation. Conversely, if a given read operation takes less time than is desirable, for instance less than approximately 300 ms, the read block for the next read operation is made larger in order to allow more data to be transferred during the next read operation. Upon the completion of the read operation, the application then relinquishes its control of the CPU returning CPU control to the operating system whereupon the system may perform another task such as reacting to a command entered by the user via the keyboard or mouse. When the operating system next returns CPU control to the application employing the method of the present invention, the read operation as described above, including dynamic sizing of read blocks, is repeated until all of the data to be transferred is read from the source I/O device or memory or until the transfer buffer is full, whichever occurs first. According, after every such read operation control of the CPU is returned to the operating system thereby preventing the user from being unduly "locked-out".

Whenever the transfer buffer is full or the last of all of the data to be transferred has been read into the transfer buffer, the method of the present invention performs a write operation using a scheme essentially similar to the data read operation described above for data read operation.

Blocks of data are transferred from the transfer buffer to a target I/O device. An initial block of data of a preselected size is written from the transfer buffer. The size of the initial write block is preselected according to the estimated speed of the type of I/O device being written to. Traditionally slow target I/O devices are initially assigned small write blocks. Traditionally fast target I/O devices are initially assigned large write blocks. For every write operation, the time required to transfer a block of data from the transfer buffer to the target I/O device is determined. If a given write operation takes longer than is desirable, for example longer than approximately 700 ms, the write block is made smaller for the next write operation in order to reduce the time required for the next write operation. Conversely, if a given write operation takes less than is desirable, for example less than approximately 300 ms, the write block is made larger for the next write operation in order to allow more data to be transferred during the next write operation. The application then relinquishes its control of the CPU returning CPU control to the operating system. When the operating system returns CPU control to the application employing the method of the present invention, the write operation described hereinabove is repeated until the transfer buffer is empty. Whenev r the

transfer buffer is empty and more source data remains to be retrieved from the source I/O device, the read operations described hereinabove are repeated. After each write operation, control of the CPU is returned to the operating system thereby preventing the user from being unduly "locked-out".

Where the time required for a read or write operation falls between the threshold values, in the present embodiment, 300 ms and 700 ms, the size of the data block to be transferred in the next operation is not altered. In an embodiment employing high and low threshold values, as opposed to employing a single threshold value, the size of the data block is not resized with every operation. A single threshold value may have application as described below.

A feature of the present invention is the ability to select an initial block size for the read operation which size is a function of the I/O source device and an initial block size for the write operation which size is a function of the I/O target device wherein the initial sizes are independent of one another. Moreover, the predetermined size of the block for the initial read or write operation may be customized for specifically identified I/O devices.

By transferring data in this manner, an application employing the method of the present invention will not 'monopolize' CPU time when transferring data to or from slow I/O devices. The use of dynamically shortened data blocks for transfers to or from slow I/O devices assures that other applications running concurrently, as well as the operating system, will be available more frequently to respond to user input. Similarly, an application employing

this method will not burden the operating system with excessive task-switching overhead time when transferring data to or from fast I/O devices. The use of dynamically enlarged data blocks to transfer data to or from fast I/O devices assures that an application employing the method of the present invention does not obtain and then relinquish CPU control too frequently when transferring large amounts of data.

An example of a data transfer by an application employing the method of the present invention will further illustrate operation of the method comprising the present invention. For this example a data file with a size of 50 Kilobytes (K) of data is to be transferred from a hard disk drive to a floppy disk drive. Since source I/O device, identified as a hard disk drive, is expected to be a relatively fast device, the block size for the first read operation will initially be set to a relatively large size of 32K. Since the target I/O device, identified as a floppy disk drive, is expected to be a slow device, the write block size will initially be set to a relatively small size of 4K. The transfer buffer in this example is of a fixed size of 64K and comprises a memory space of 64K allocated in the computer system random access memory (RAM). High and low threshold values for read/write access times are set at 700 ms and 300 ms, respectively.

The application employing the method of the present invention first initiates a read of 32K, the initial size of the read data block, transferring 32K of data from the hard disk drive into the transfer buffer. Upon the compl tion of this disk access, the 32K of data from the hard disk is located in the transfer buffer. The time taken to perform

the read operation is determined using the computer system "tick counter". If, for example, this read operation took 2 seconds, the read buffer is dynamically shortened for the next read of the hard disk drive, for instance by halving its size to 16K, since this read operation was slower than the 700 ms threshold. The application then relinquishes control of the CPU.

Upon reactivation of the application and its transfer task by the return of CPU control from the operating system, an additional 16K of data will be read from the hard disk drive into the transfer buffer. If, for example, this read access took 500 ms, the read buffer will not be resized, since this access time is between the two threshold values. Control of the CPU is then relinquished by the application.

Upon the next activation of the application, the remaining 2K of the 50K data file is read from the hard disk drive into the transfer buffer, and control of the CPU is relinquished by the application.

Upon reactivation of the application, 4K of data, the initial size of the write block, is transferred from the transfer buffer to the floppy disk drive. If, for example, this write operation took 100 ms, the write block for the next operation is dynamically resized, for instance by doubling its size to 8K, since this write operation was faster than the 300 ms threshold. Control of the CPU is then relinquished by the application.

Upon the next activation of the application, an 8K block of data is transferred from the transfer buffer to the floppy disk drive. If, for example, this access took 150 ms, the write block for the next operation is dynamically resized, for instance

by doubling its size to 16K, since this write operation was faster than the 300 ms threshold. The application then releases its control of the CPU.

Upon reactivation of the application, a 16K block of data is transferred from the transfer buffer to the floppy disk drive. If, for example, this write operation took 200 ms, the write block for the next write operation is dynamically resized, for instance by doubling its size to 32K, since this access was faster than the 300 ms threshold. Control of the CPU is relinquished by the application.

Upon the next activation of the task, a 22K block of data, the amount of data remaining in the transfer buffer, is transferred to the floppy disk drive. This write operation completes the data transfer of this example. The application then relinquishes CPU control. For further data transfers involving the hard disk drive or floppy disk drive, the current buffer sizes of 16K and 32K, respectively, are utilized for initial read or write operations, rather than the default sizes originally employed.

Brief Description Of The Drawing

Fig. 1 is an illustration of the path through which data is transferred by the method comprising the present invention.

Best Mode For Carrying Out The Invention

While this invention is susceptible of embodiment in many different forms, there is shown in the drawings and will herein be described in detail, several specific embodiments, with the understanding that the present disclosure is to be considered as an exemplification of the principles of the invention and is not intended to limit the invention to the embodiments illustrated.

The path through which data is transferred from a source I/O device to a target I/O device by the method comprising the present invention is illustrated in FIG. 1. Blocks of data, (read blocks), are read from source I/O device 10, via data link 13, into transfer buffer 11. Blocks of data, (write blocks), are written from transfer buffer 11, via data link 14, to target I/O device 12.

Source I/O device 10 may comprise computer memory, or any device which may be interfaced to a computer system to provide a source of data to a computer system, such as one or more hard disk drives, floppy disk drives, network file servers, CD-ROM drives, modems, tape drives, scanners, etc. Different source I/O devices may be selected to source data at different times. Source I/O device 10, as depicted in FIG. 1, is thus a representation of the currently selected source of data, being one of many data sources available within the computer system.

Similarly, target I/O device 12 may comprise computer memory, or any device which may be interfaced to receive data from a computer system, such as one or more hard disk drives, floppy disk drives, network file servers, printers, modems, tape drives, etc. Different target I/O devices may be selected

to receive data at different times. Source I/O device 12, as depicted in FIG. 1, is thus a representation of the currently selected target of data, being one of many data targets available within the computer system.

Blocks of data read from source I/O device 10 are transferred to and within the computer system, via data link 13, to transfer buffer 11. Data link 13, such as formed by the combination of an address and a a data bus, allows numerous source I/O devices to be simultaneously connected to transfer buffer 11.

Blocks of data written to target I/O device 12 are transferred within and from the computer system, via data link 14, from transfer buffer 11. Data link 14, such as formed by the combination of an address and a a data bus, allows numerous target I/O devices to be simultaneously connected to transfer buffer 11.

Transfer buffer 11 is a fixed-sized, contiguous portion of memory allocated within the computer system's RAM. The size of transfer buffer 11 may be selected by a developer of an application employing the method of this invention, based upon available memory resources. A relatively large transfer buffer size, such as 64K, may allow several consecutive data access to or from the same I/O device to occur relatively closely together in time. This frequently can result in more efficient data transfer operations, since two consecutive data accesses to or from an I/O device that occur relatively closely together in time frequently takes less net access time than two data accesses that are spaced apart in time. This may occur, for example, when two data blocks to be accessed are located within the same general area on the surface media of a

floppy diskette. In such a case, the time to bring the floppy diskette up to the proper rotational speed, as well as the time to move the read/write head of the floppy disk drive from a 'parked' position, may be 'shared' among the two consecutive data accesses, rather than being separately incurred by each data access when the accesses are spaced apart in time. For an IBM compatible computer interfaced to typical hard disk drives and floppy disk drives, a transfer buffer size of 64K has been determined to provide adequate performance.

The read blocks employed to transfer data from source I/O device 10 to transfer buffer 11 are dynamically sized. The write blocks employed to transfer data from transfer buffer 11 to target I/O device 12 are also dynamically sized. Prior to the first initiation of a data transfer utilizing particular source and target I/O devices, the sizes of the read and write blocks are initially set based upon the estimated speed of the particular type of source and target I/O devices, respectively. Relative small read or write blocks are initially assigned for traditionally slow I/O devices, such as floppy disk drives. After every data access operation, the application employing the method of this invention will relinquish its control of the CPU. Since the time required to perform a read or write data access is relatively proportional to the amount of data being transferred, and since data accesses cannot be interrupted in midstream by an application running under a nonpreemptive multitasking operating system, the use of a small read or write block for a traditionally slow I/O device will assure that the application will relinquish CPU control relatively quickly, rather than monopolize CPU time.

Similarly, relatively large data blocks are initially employed for traditionally fast I/O d vices, such as hard disk drives. Since task-switching overhead occurs every time the application relinquishes or regains CPU control, the use of large blocks for a traditionally fast I/O device will assure that the application does not relinquish CPU control too frequently, which could result in degradation of overall system performance.

These initial read and write block size values may be selected by a developer of an application employing the method of the present invention based upon the estimated speeds of I/O devices to be employed within a particular computer system. An initial block size of 32K has been determined to perform adequately for many hard disk drives. An initial block size of 16K has been determined to perform adequately for a many hard disk drives accessed by the computer system via a local area network. An initial block size of 4K has been determined to perform adequately for many floppy disk drives.

The various embodiments of the invention described herein assign the same initial block size to a particular type of I/O device regardless of whether the particular device is a source I/O device or a target I/O device, and thus regardless of whether a read block or a write block is used to access a particular device. Identical initial values are employed because devices with both read and write access capability, such as hard disk drives, generally have similar read and write access times. The present invention further contemplates that one could easily employ different initial for a particular type of I/O device for read and write block

sizes, should a particular I/O device with markedly disparate read and write access times be encountered.

Initial read and write block sizes, based upon the estimated speed of particular types of I/O devices, may not accurately reflect the amount of time actually required to read or write a block of data of a given size from or to an actual I/O device. Moreover, the access time of a given I/O device may vary over time. For example, the fragmentation of data files that may occur over time within a hard disk drive may result in a corresponding increase in access time, since consecutively accessed data are less likely to be located within adjacent sectors and tracks on the hard disk's media surface. Further, the time to access a file server accessed via a local area network may vary over time depending upon overall loading of the network. Accordingly, the method described by the present invention comprises the reading and writing of data blocks which are dynamically sized to assure that applications utilizing this method will maintain CPU control for relatively uniform periods of time, regardless of the speed or type of I/O devices it accesses.

For each read or write access of a source or target I/O device, respectively, by the application, the time required to complete the access is determined. On an IBM compatible personal computer, this may be accomplished by interrogating the computer's clock tick counter, which is incremented 18.2 times per second. The clock tick counter, maintained by the computer's ROM BIOS, is documented within "Advanced MS-DOS Programming", p. 589, by Ray Duncan, published in 1988 by Microsoft Press of Redmond, Washington. By obtaining the values of the

clock tick counter both before and after an I/O device data access and then subtracting the difference between the two values, the time required to perform the data access can be calculated.

The calculated data access time for a read or write operation of an I/O device is used to dynamically size the read or write data block for the next access of the same I/O device. Several methods of dynamic data block sizing, all within the general scope of this invention, may be employed. Three embodiments of such dynamic data block methods are described hereinbelow.

In a first embodiment of the present invention, the calculated data access time for a read or write operation, determined as described hereinabove, is then compared to two threshold values; an upper threshold and a lower threshold. These threshold values are predetermined, and are not varied by the application. These threshold values may be selected by a developer of an application employing the method of this invention in order to achieve optimal system performance and responsiveness to user input for the particular computer, operating system, and concurrently running applications that comprise the environment in which the application is running. For an IBM compatible computer with a 386 or 486 class microprocessor running MS-DOS 5.0 and Windows 3.1, an upper threshold of approximately 700 ms and a lower threshold of approximately 300 ms second have been determined to provide adequate performance.

If the calculated access time is greater than or equal to the upper threshold value, the data access is deemed to have taken too long. In order to shorten the time r quired for the next read or write

access of the same source or target I/O device, the read or write data block is dynamically shortened by halving its size. A read or write data block will not be dynamically shortened, however, if it has already been dynamically shortened to a set minimum size, such as 1K. Since CPU control is relinquished by the application after each I/O device access, and since successive I/O accesses to the same I/O device will often take approximately the same amount of time, the application will likely not maintain CPU control for too much time during the next read or write access of the same I/O device, when the dynamically shortened read or write buffer is employed.

If the calculated access time is less than or equal to the lower threshold value, the data access is deemed to have taken too little time. In order to expand the time required for the next read or write access of the same source or target I/O device, the read or write block is enlarged by doubling its size. The read or write block will not be dynamically enlarged, however, if it has already been dynamically expanded to a set maximum size, such as the size of the transfer buffer. Enlarging the read or write block will allow more data to be transferred from or to the same I/O device the next time CPU control is returned to the application. This will allow increased operating system responsiveness, since the operating system will perform fewer task switches, and will thus incur less task-switching overhead time.

If the calculated access time is less than the upper threshold value and greater than the lower threshold value, the data access is deemed to have taken approximately the correct amount of time. The read or write block is not dynamically resized.

Instead, the same size read or write block is employed for the next read or write access of the same I/O device.

In a second embodiment of the method of the present invention, upper and lower threshold values are not used, and data blocks are not sized by halving or doubling their current sizes. Instead, a uniform data access time is determined. The uniform data access time is a preset, fixed value, and may be selected in order to achieve optimal system performance and responsiveness to user input for the particular computer, operating system, and concurrently running applications that comprise the environment in which the application is running. For an IBM compatible computer with a 386 or 486 class microprocessor running MS-DOS 5.0 and Windows 3.1, a uniform data access time of approximately 500 ms has been determined to provide adequate performance.

After each read or write operation from or to an I/O device, the current size of the read or write block is divided by the calculated access time, determined as described hereinabove, to determine the rate, in units such as kilobytes per millisecond, at which the data access actually took place. This rate is then multiplied by the uniform data access time. The product of this multiplication, in units such as kilobytes, is the new data block size for the next read or write operation from the same I/O device. If the new data block size is smaller than a preset minimum value, such as 1K, the new data block size is set to the preset minimum value. If the new data block size is greater than a preset maximum value, such as the size of the transfer buffer, then the new data block size is set to the preset maximum value.

By dynamically sizing the data buffer in this manner, should the next access of the same I/O device occur at the same rate as the most recent access, the next access will take exactly the uniform access time. This method of dynamic buffer sizing will often achieve a more constant data access time than the first preferred embodiment of this invention, and will often achieve such uniformity more quickly.

A third embodiment of the present invention is designed to perform more efficiently with block oriented I/O devices commonly interfaced to IBM compatible computer systems. Block oriented I/O devices, such as hard disk drives commonly connected to IBM compatible computers, operate most efficiently when the size of the blocks of data transferred to and from the device are a power of two, such as 4K. Accordingly, the dynamically sized data blocks are adjusted so that their size is a multiple of 4K. This embodiment of the present invention also adjusts the dynamically sized data blocks so that they are a simple factor, or simple fraction, of the overall transfer buffer size. This can result in more efficient data transfer operations, since multiple data blocks within the transfer buffer will not likely have markedly different sizes.

After each read or write operation from or to an I/O device, the amount of data most recently read or written from or to the I/O device is adjusted by rounding its size up to the next highest multiple of 4K. For example, if 4097 bytes were most recently read or written, the amount of data is set at 8K. This amount of data is divided by the sum of the calculated access time, described hereinabove, plus a constant value. The constant value prevents the

divisor from being an extremely small value, and thus prevents extremely large adjustments to data block sizes. A constant value of one fourth of the fixed transfer buffer size has been determined to perform adequately. The result of this division approximates the rate, in units such as kilobytes per millisecond, at which the data access actually took place. This rate is then multiplied by the desired data access time. The product of this multiplication, in units such as kilobytes, is the preliminary new data block size for the next read or write operation from or to the same I/O device.

This preliminary data block size is then adjusted so that it is close to a simple factor of the size of the transfer buffer. The size of the transfer buffer is divided, using integer arithmetic, by the preliminary block size. This yields the number of entire preliminary data blocks that fit inside the transfer buffer. The size of the transfer buffer is then divided, again using integer arithmetic, by the number of entire preliminary data blocks that fit inside the transfer buffer. This yields a new data block size that is close to a simple factor of the transfer buffer size. Finally, this new data block size is adjusted, if necessary, by rounding its size up to the next highest multiple of 4K. If the new data block size is smaller than a preset minimum value, such as 4K, the new data block size is set to the preset minimum value. If the new data block size is greater than a preset maximum value, such as the size of the transfer buffer, then the new data block size is set to the preset maximum value.

Regardless of which preferred method described hereinabove is employed to dynamically size data blocks, the general process employed by the method

of the present invention for transferring data remains the same.

The foregoing description and drawing merely explain and illustrate the invention and the invention is not limited thereto, except in so far as the appended claims are so limited, as those skilled in the art who have the disclosure before them will be able to make modifications and variations therein without departing from the scope of the invention.

Claims

1. A method for transferring data between Input/Output devices interfaced to a computer system utilizing a nonpreemptive multitasking operating system, said computer system comprising a CPU and random access memory, said method for transferring data between Input/Output devices comprising the steps of:

- A. defining a transfer buffer within the random access memory;
- B. setting an initial read block size;
- C. setting an initial write block size;
- D. reading a block of data from said source Input/Output device into said transfer buffer;
- E. determining the amount of time required to read the block of data from said source Input/Output device into said transfer buffer;
- F. resizing said read block for the next read of data from said source Input/Output device;
- G. relinquishing control of the CPU to the operating system;
- H. repeating steps D through G, upon reacquiring control of said CPU, until either said transfer buffer is full or all data to be transferred from said source Input/Output device to said target Input/Output device has been read from said source Input/Output device;
- I. writing a block of data from said transfer buffer to said target Input/Output device;
- J. determining the amount of time required to write said block of data from said transfer buffer to said target Input/Output device;
- K. resizing said write block for the next write of data to said target Input/Output device;

- L. relinquishing control of said CPU to said operating system;

- M. repeating steps I through L, upon reacquiring control of said CPU, until said transfer buffer is empty; and

- N. repeating steps D through M until all data has been transferred from said source Input/Output device to said target Input/Output device.

2. The invention according to Claim 1 in which the step of setting an initial read block size is based upon an estimated speed of a source Input/Output device.

3. The invention according to Claim 1 in which the step of setting an initial write block size based upon an estimated speed of a target Input/Output device.

4. The invention according to Claim 1 in which the step of determining the amount of time required to read the block of data from said source Input/Output device into said transfer buffer further comprises:

- A. interrogating said computer system clock tick counter prior to the step of reading a block of data;

- B. interrogating said computer system clock tick counter immediately after the step of reading a block of data; and

- C. subtracting the value of said computer system clock tick counter immediately before the step of reading a block of data from the value of said computer system clock tick counter immediately after the step of reading a block of data, whereby the difference between said values corresponds to the time required to read said block of data from said source Input/Output device into said transfer buff-

er.

5. The invention according to Claim 1 in which the step of determining the amount of time required to write the block of data from said transfer buffer to said target Input/Output device further comprises:

- A. interrogating said computer system clock tick counter prior to the step of writing a block of data;

- B. interrogating said computer system clock tick counter immediately after the step of writing a block of data; and

- C. subtracting the value of said computer system clock tick counter immediately before the step of writing a block of data from the value of said computer system clock tick counter immediately after the step of writing a block of data, whereby the difference between said values corresponds to the time required to write said block of data from said from said transfer buffer to said target Input/Output device into said transfer buffer.

6. The invention according to Claim 1 in which the step of resizing said read block for the next read of data from said source Input/Output device further comprises the steps of:

A. predetermining an upper threshold value above which the time required to read a block of data from said source Input/Output device is deemed to have taken too long;

B. predetermining a lower threshold value below which the time required to read a block of data from said source Input/Output device is deemed to have taken too little time; and

C. alternatively,

- increasing the size of said read block for

the next read of data in the event that the time required to read a block of data from said source Input/Output device is equal to or less than said lower threshold value, or

- decreasing the size of said read block for the next read of data in the event that the time required to read a block of data from said source Input/Output device is equal to or greater than said upper threshold value.

7. The invention according to Claim 6 wherein the size of said read block is doubled in the event that the time required to read a block of data from said source Input/Output device is equal to or less than said lower threshold value.

8. The invention according to Claim 6 wherein the size of said read block is halved in the event that the time required to read a block of data from said source Input/Output device is equal to or greater than said upper threshold value.

9. The invention according to Claim 6 wherein the size of said write block is doubled in the event that the time required to write a block of data to said target Input/Output device is equal to or less than said lower threshold value.

10. The invention according to Claim 6 wherein the size of said write block is halved in the event that the time required to write a block of data to said target Input/Output device is equal to or greater than said upper threshold value.

11. The invention according to Claim 1 in which the step of resizing said read block for the next read operation of data from said source Input/Output device further comprises the steps of:

A. predetermining a uniform access time corresponding to the preferred time required to read a

block of data from said source Input/Output device;

B. determining the rate at which data from said source Input/Output device is read; and

C. determining a new read block size for the next read operation from said source Input/Output device by multiplying said rate by said uniform data access time.

12. The invention according to Claim 1 in which the step of resizing said write block for the next write operation of data to said target Input/Output device further comprises the steps of:

A. predetermining a uniform access time corresponding to the preferred time required to write a block of data to said target Input/Output device;

B. determining the rate at which data from said target Input/Output device is written; and

C. determining a new write block size for the next write operation to said target Input/Output device by multiplying said rate by said uniform data access time.

13. The invention according to Claim 1 in which the step of resizing said read block for the next read operation of data from said source Input/Output device further comprises the steps of:

A. predetermining a uniform access time corresponding to the preferred time required to read a block of data from said source Input/Output device;

B. determining a value which is a power of two and a suitable size for reading blocks of data from a block orientated source Input/Output device;

C. rounding the amount of data most recently read from said source Input/Output device to the next highest multiple of said power of two.

D. determining the approximate rate at which data from said source Input/Output device is read;

E. determining a preliminary new read block size;

F. determining the number of said preliminary new read blocks that fit within said transfer buffer; and

G. determining a new read block size for the next read operation from the same source Input/Output device.

14. The invention according to Claim 13 in which the step of determining a preliminary new read block size further comprises the step of dividing the size of said transfer buffer by said approximate rate.

15. The invention according to Claim 13 in which the step of determining the number of said preliminary new read blocks that fit within said transfer buffer further comprises the step of dividing the size of said transfer buffer by said preliminary new read block size.

16. The invention according to Claim 13 in which the step of determining a new read block size for the next read operation from the same source Input/Output device further comprises the step of dividing the size of said transfer buffer by said number of entire said preliminary new read blocks that fit within said transfer buffer.

17. The invention according to Claim 1 in which the step of resizing said write block for the next write operation of data to said target Input/Output device further comprises the steps of:

A. predetermining a uniform access time corresponding to the preferred time required to write a block of data to said target Input/Output device;

B. determining a value which is a power of two and a suitable size for writing blocks of data to a block orientated target Input/Output device;

C. rounding the amount of data most recently

written to said target Input/Output device to the next highest multiple of said power of two.

D. determining the approximate rate at which data was written to said target Input/Output device;

E. determining a preliminary new write block size;

F. determining the number of said preliminary new write blocks that fit within said transfer buffer; and

G. determining a new write block size for the next write operation to the same target Input/Output device.

18. The invention according to Claim 17 in which the step of determining a preliminary new write block size further comprises the step of dividing the size of said transfer buffer by said approximate rate.

19. The invention according to Claim 17 in which the step of determining the number of said preliminary new write blocks that fit within said transfer buffer further comprises the step of dividing the size of said transfer buffer by said preliminary new write block size.

20. The invention according to Claim 17 in which the step of determining a new write block size for the next write operation to the same target Input/Output device further comprises the step of dividing the size of said transfer buffer by said number of entire said preliminary new write blocks that fit within said transfer buffer.

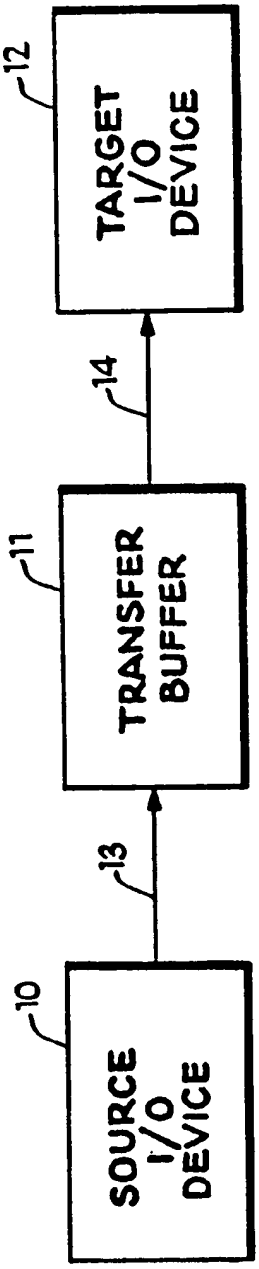

SOURCE I/O DEVICE
10
13
TRANSFER BUFFER
11
14
TARGET I/O DEVICE
12

Fig 1